

*Enterprise System  
Integration with Web  
Services*

*Dr. Lixin Tao*

Computer Science Department  
Pace University

# Outline

---

- Enterprise System Integration: Key for Business Success
- Key Challenges to Enterprise System Integration
- Service-Oriented Architecture (SOA)
- Alternative Technologies for Enterprise System Integration
- Web Architecture Review
- Web Service Architecture and Composition
- Developing and Consuming Web Services
- Web Service Security and Performance
- Conclusion

# Enterprise System Integration: Key for Business Success

---

- A company's business process is normally supported by various information systems based on different technologies
- Major companies have operating units distributed across different cities or countries
- No company can be self-sufficient. Each company plays the roles of both consumer and producer
- It is more cost-effective to buy services from domain experts
- **Conclusion:** *Both internal and external information systems need be integrated efficiently*

# Key Challenges to Enterprise System Integration

---

- They may run on heterogeneous hardware/OS platforms
- They may be implemented in different programming languages
- They may be based on various software frameworks like J2EE and .NET, or other proprietary techniques
- Most systems are protected by firewalls, and there are security concerns in opening special ports
- **Question:** *How can heterogeneous information systems interact with each other?*

# Service-Oriented Architecture (SOA)

---

- SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents
- A service is a unit of work done by a service provider to achieve desired end results for a service consumer.
- Both provider and consumer are roles played by software agents on behalf of their owners
- *Example:* An online book broker store, like <http://www.bookfinder4u.com>
  - B2C: The broker provides service of finding the best book prices
  - B2B: The broker's IS checks major bookstore's IS services for their book prices

# Service-Oriented Architecture (SOA) ...

---

- The success of SOA depends on
  - A standard interfacing technology adopted by all participating software agents
  - Consumers don't need to install special tools to use each special service
  - A mechanism for consumers to find potential service providers
  - Consumers can easily switch to more cost-effective providers for similar services
- **Observation:** *Standardization of interfacing technology and business service API are critical*

# Outline

---

- Enterprise System Integration: Key for Business Success
- Key Challenges to Enterprise System Integration
- Service-Oriented Architecture (SOA)
- **Alternative Technologies for Enterprise System Integration**
- Web Architecture Review
- Web Service Architecture and Composition
- Developing and Consuming Web Services
- Web Service Security and Performance
- Conclusion

# Alternative Technologies for Enterprise System Integration

---

- **CORBA**

- Developed by *OMG* (<http://omg.org>) with over 200 IT companies in early 1990s
- Supporting system integration based on heterogeneous platforms and languages
- Not successful in business due to its complexity and compromising diverse views

# Alternative Technologies for Enterprise System Integration ...

---

- J2EE
  - Based on industry standard embraced by most IT companies except Microsoft
  - Java RMI-IIOP is based on CORBA
  - Efficient solution if all participating software agents are Java-based

# Alternative Technologies for Enterprise System Integration ...

---

- **Microsoft .NET**
  - Proprietary technology
  - Base on mature COM/DCOM/COM+ technology
  - Efficient solution if all participating software agents are Windows-based

# Alternative Technologies for Enterprise System Integration ...

---

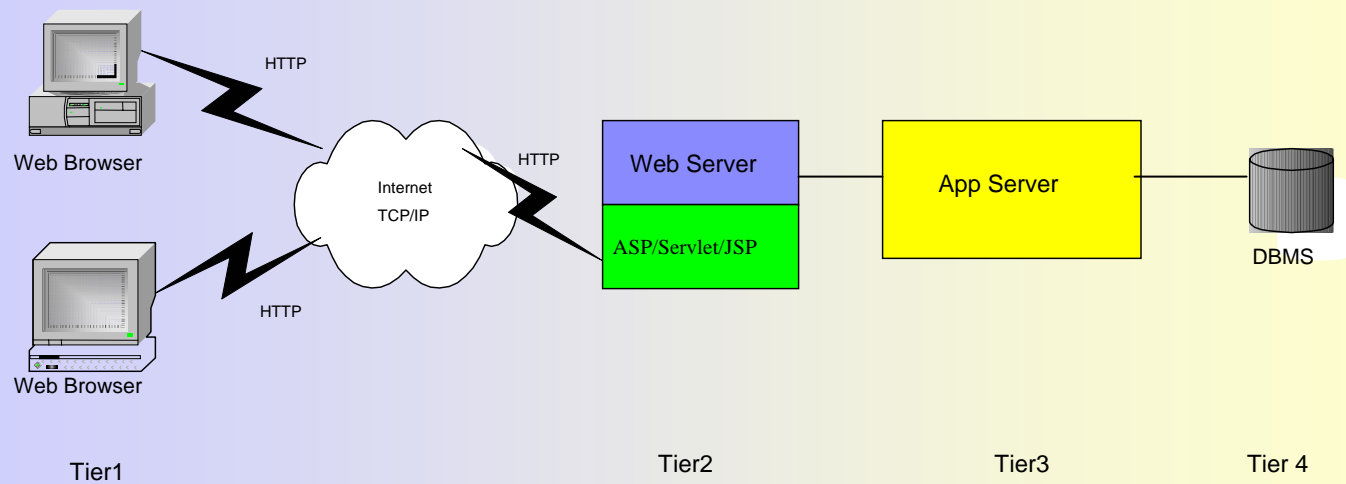
- Web services
  - Non-proprietary, based on industry standards
  - Support SOA with XML and Internet protocols
  - Like CORBA, supporting system integration based on heterogeneous platforms and languages
  - Embraced by most IT companies, and *Microsoft* is one of its major proponents
  - Convenient for transforming legacy applications into service providers
  - Limited functionality, still evolving
  - Mainly suitable for loose coupling due to its limited performance

# Outline

---

- Enterprise System Integration: Key for Business Success
- Key Challenges to Enterprise System Integration
- Service-Oriented Architecture (SOA)
- Alternative Technologies for Enterprise System Integration
- **Web Architecture Review**
- Web Service Architecture and Composition
- Developing and Consuming Web Services
- Web Service Security and Performance
- Conclusion

# The Four-Tiered Web Architecture



Server presentation tier: CGI, Servlet, JSP, ASP

Business logic tier: EJB, COM+, CORBA

# HTTP Protocol Basics

---

- A simple application-level hand-shaking rule between a client (*Web browser or program*) and a Web server
- Each Web server component has a unique *URL* like ***http://csis.pace.edu:80/survey/input***
- The client opens a TCP/IP connection with a Web server, sends an HTTP *Get* or *Post* request to one of its Web components (say, /survey/input)
- The Web component processes the data submitted by the client, and returns an *HTTP response* to the client

# HTTP Protocol Basics ...

---

- Sending HTTP request with an HTML form

```
<html>
<body>
  <form method= "post"  action="http://csis.pace.edu/survey/input">
    Enter your name: <input type="text"  name="user">
    <input type="submit"  value="Submit">
  </form>
</body>
</html>
```

Enter your name:

Submit

# HTTP Protocol Basics ...

---

- Sample HTTP POST request

```
POST /survey/input HTTP/1.0
Accept: text/html
Accept: audio/x
User-agent: MacWeb

user=Ada
```

(Assume the user typed “Ada” in the textbox)

- *Observation:* The *entity body* can carry any text or binary data

# HTTP Protocol Basics ...

---

- Sample HTTP response

```
HTTP/1.0 200 OK
Server: NCSA/1.3
Mime_version: 1.0
Content_type: text/html
Content_length: 2000
```

```
<HTML>
```

```
.....
```

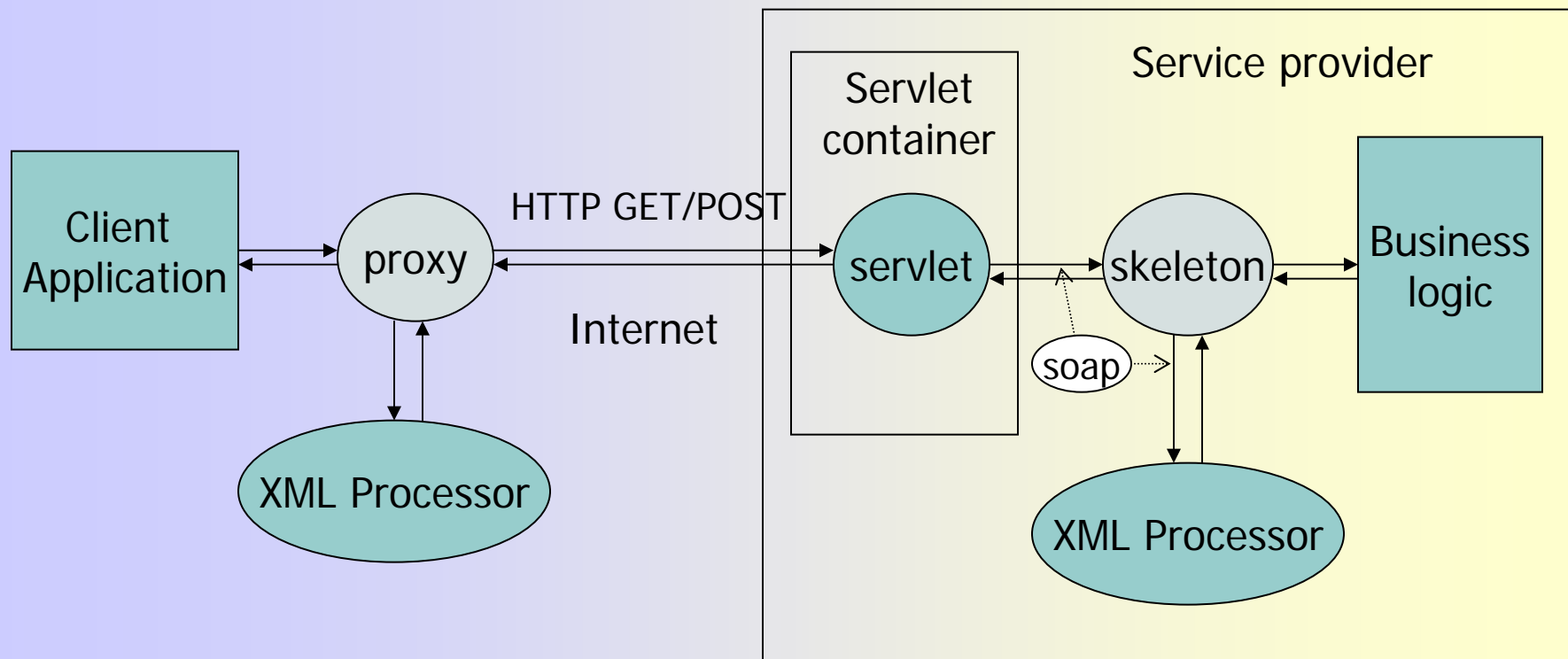
```
</HTML>
```

# Outline

---

- Enterprise System Integration: Key for Business Success
- Key Challenges to Enterprise System Integration
- Service-Oriented Architecture (SOA)
- Alternative Technologies for Enterprise System Integration
- Web Architecture Review
- **Web Service Architecture and Composition**
- Developing and Consuming Web Services
- Web Service Security and Performance
- Conclusion

# Fundamental Concepts for Web services



# Fundamental Concepts for Web services ...

---

- *Web service* is a method on the hosting server that is open for client programs to invoke
- The signature of the publicized method is described by a *Web Service Definition Language (WSDL)* file, which is automatically generated by a tool
- A client may get the Web service WSDL file from a *UDDI* registry or directly over the Web, and a tool will transform the WSDL file into a proxy class, which exposes the same method signature as that on the server

# Fundamental Concepts for Web services ...

---

- When the client calls the proxy's method, its body generates a *SOAP* message representing all information about the method call, and then uses *HTTP POST request* to send the SOAP file to a Web component (servlet or ASP) also automatically generated by a tool
- The Web component transforms the SOAP message into a method call to the local Web service implementation method

## Fundamental Concepts for Web services ...

---

- Upon receiving method return value, the Web component transforms it into a *SOAP response* message, and send it back to the proxy object as part of its *HTTP response*
- The proxy method body parses the *SOAP response* message into a value in its own language, and return it as its own return value to the client

# Web Service Definition Language (WSDL)

---

- XML representation of all information about Web service methods' signatures, as well as the URL for their entry-point Web component
  - Method name
  - Parameters and their data types
  - Return type
  - XML Schema representation of above data types if they are user-defined, as well as *Exceptions* that the Web service implementation may throw
  - URL for the entry-point Web component

# Sample WSDL File

---

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions>
  <message name="squareResponse">
    <part name="squareReturn" type="xsd:int" />
  </message>
  <message name="squareRequest">
    <part name="x" type="xsd:int" />
  </message>
  <service name="SquareIntegerServerService">
    <port binding="SquareIntegerServerSoapBinding"
      name="SquareIntegerServer">
      <address
        location="http://localhost:8080/axis/SquareIntegerServer.jws" />
    </port>
  </service>
</definition>
```

# Simple Object Access Protocol (SOAP)

---

- XML representation of all information about a method call and its return value
- SOAP message's service-specific structure is defined by WSDL
  - Method name
  - Method argument values
  - Method return value
  - Method exceptions

# Sample SOAP Message

---

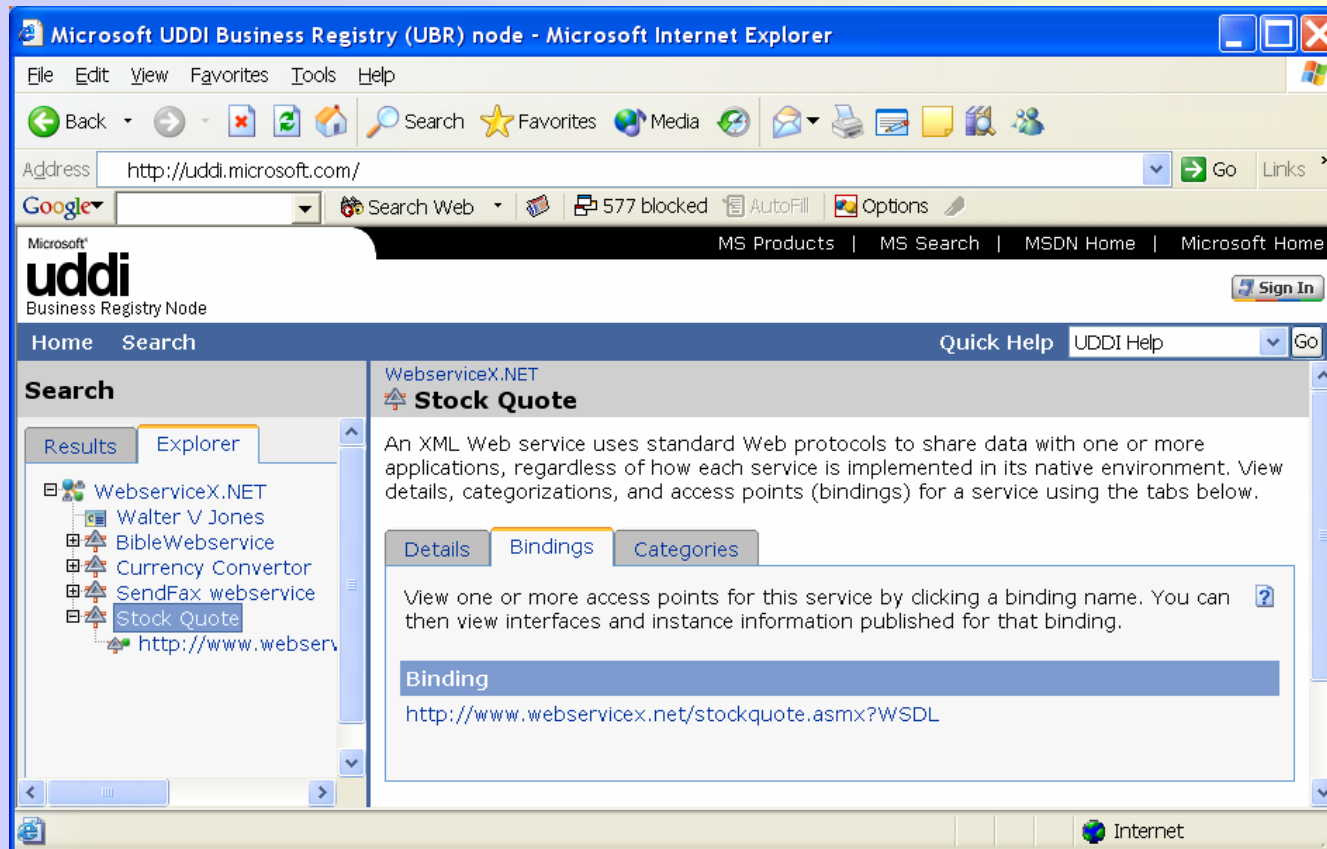
```
<?xml version='1.0' ?>  
<Envelope>  
  <Body>  
    <symbolist>  
      <symbol>C</symbol>  
      <symbol>GE</symbol>  
      <symbol>DJI</symbol>  
    </symbolist>  
  </Body>  
</Envelope>
```

# Universal Description, Discovery and Integration (UDDI)

---

- A standard for Web service registries
  - Web service providers can register and advertise their services in the registries
  - Potential clients can use the registries as yellow pages to search for potential service providers and download the WSDL files for the chosen services
- UDDI allows programs to access the registries through their open APIs.

# Universal Description, Discovery and Integration (UDDI) ...



# Outline

---

- Enterprise System Integration: Key for Business Success
- Key Challenges to Enterprise System Integration
- Service-Oriented Architecture (SOA)
- Alternative Technologies for Enterprise System Integration
- Web Architecture Review
- Web Service Architecture and Composition
- **Developing and Consuming Web Services**
- Web Service Security and Performance
- Conclusion

# Basic Steps for Developing Web Services

---

- Identify the signatures of public methods to be published as Web services
  - Creating a Java interface file for the signatures
  - Mark such methods with “WebMethod” attribute (.NET)
- Use a tool to generate a WSDL file and a Web component for the new Web services based on the chosen method signatures and a Web server URL
- Deploy the Web component (and WSDL file) on the Web server
- Publish the WSDL file and business descriptions for the Web services in a UDDI registry (optional)

# Basic Steps for Developing Web Services ...

---

- Java example

```
public interface HelloService {  
    String hello(String name);  
}
```

- C# .NET example

```
[WebService(Namespace="http://csis.pace.edu/")]  
public class HelloService : System.Web.Services.WebService {  
    [WebMethod]  
    public string Hello(string name) {  
        return "Hello World, " + name + "!";  
    }  
}
```

# Basic Steps to Consume a Web Service

---

- Find WSDL file URL for the Web service
- Use a tool to download the WSDL file and generate the supporting source files for the proxy class in a language of your choice
- In your client application, create a proxy object (maybe through another proxy factory object), and treat the proxy object as if it is the Web service implementation

# Basic Steps to Consume a Web Service ...

---

- Java example

```
public class HelloClient {
    public static void main(String[] args) throws Exception {
        String name = args[0];
        // Create a proxy factory object
        HelloServiceLocator factory = new HelloServiceLocator();
        // Generate a Web service proxy object
        HelloService proxy = factory.getHelloService();
        // Invoke Web service method
        String message = proxy.hello(name);
        System.out.println(message);
    }
}
```

# Basic Steps to Consume a Web Service ...

---

- C# .NET example

```
public class HelloClient
{
    string Hello(string name) {
        HelloService proxy = new HelloService();
        return proxy.Hello(name);
    }
}
```

# Outline

---

- Enterprise System Integration: Key for Business Success
- Key Challenges to Enterprise System Integration
- Service-Oriented Architecture (SOA)
- Alternative Technologies for Enterprise System Integration
- Web Architecture Review
- Web Service Architecture and Composition
- Developing and Consuming Web Services
- **Web Service Security and Performance**
- Conclusion

# Web Service Security

---

- Use *Secure Socket Layer (SSL)* protocol to provide a secure communication channel between the Web services and their client systems
- Encrypt SOAP data by the client-side's proxy class, and decrypt SOAP data at the Web server component end.
- Implement authentication and authorization mechanisms at OS level or application level

# Web Service Performance

---

- Web service itself is a wrapper technology. Its layered translations and forwarding incur performance penalties
- Web service is suitable mainly for loosely-coupled computing
- Web service performance depends on the implementation technology of its business logics: using simple object? EJB? COM+?
- **Conclusion:** *Web service is supplementing application server technologies like EJB and COM+, not replacing them*

# Conclusion

---

- Industry trend is integration of distributed and specialized services based on the Service-Oriented Architecture
- Web service is an emerging technology supporting Service-Oriented Architecture
- Functioning as a common denominator of major competing technologies, instead of an aggressive panacea like CORBA, it has been quickly adopted by most major IT companies
- Web services depend on application server technologies like EJB and COM+ for performance
- OO, component-based SE, and SOA are all important in different levels of information system hierarchy; they are not replacing each other
- Web service is an important technology within reach of people with limited programming experience